Traditional Machine Learning Methods on Graphs II

Prof. O-Joun Lee

Dept. of Artificial Intelligence,
The Catholic University of Korea
ojlee@catholic.ac.kr

네트워크 과학 연구실 NETWORK SCIENCE LAB

가톨릭대학교 THE CATHOLIC UNIVERSITY OF KOREA

Contents

- Homogeneous Graph Embedding Models
  - Adjacency-based Similarity
  - Multi-hop Similarity
  - SDNE and Subgraph2Vec
- From Homogeneous to Heterogeneous Graphs
- Heterogeneous Graph Embedding Models
  - Metapath2Vec
  - HIN2Vec
  - MetaGraph2Vec
  - JUST

ReCap: Graph Representation Learning

➢ Graph Representation Learning aims to generate graph representation vectors that describe graph's structure. So we don't need to do feature engineering every single time.

Input Graph → **Structured Features** → **Learning Algorithm** → **Prediction**

❌ **Feature Engineering**

✅ **Representation Learning**

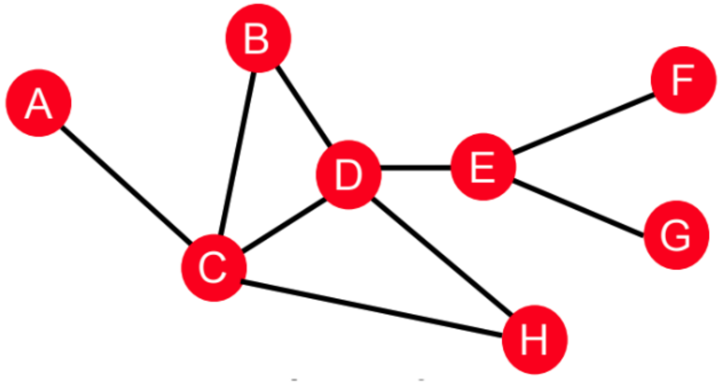
learn the features by itself

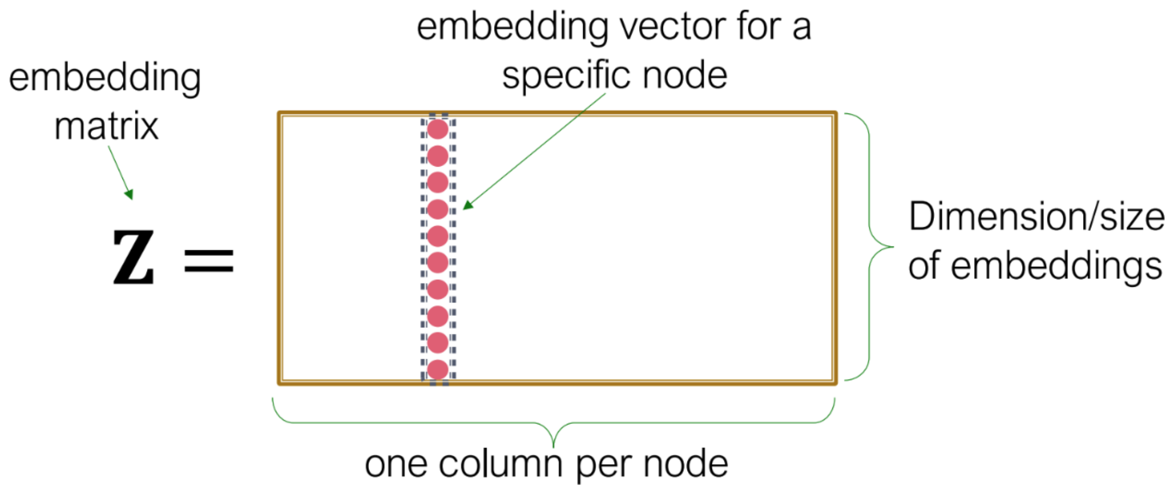
- SVM
- Random Forest
- XGBoost
- DNN

- Node-level
- Edge-level
- Graph-level

- We want to learn the embedding for every node $\mathrm{ENC}(v) = \mathbf{z}_v = \mathbf{Z} \cdot v$ such that:

$$\underbrace{\mathrm{similarity}(u, v)}_{\text{in the original network}} \approx \underbrace{\mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u}_{\text{Similarity of the embedding}}$$





How to Define Node Similarity?

- Key distinction between "shallow" methods is **how they define node similarity.**
  - E.g., should two nodes have similar embeddings if they….
  - are connected?
  - share neighbors?
  - have similar "structural roles"?
  - …

$$\underbrace{\text{similarity}(u, v)}_{\text{in the original network}} \approx \underbrace{\mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u}_{\text{Similarity of the embedding}}$$

1. Adjacency-based Similarity

- **Similarity function** is just the edge weight between *u* and *v* in the original network.
- **Intuition**: Dot products between node embeddings approximate edge existence.

$$\underbrace{\boxed{\mathcal{L}}}_{\text{loss (what we want to minimize)}} = \underbrace{\boxed{\sum_{(u,v)\in V\times V}}}_{\text{sum over all node pairs}} \| \underbrace{\boxed{\mathbf{z}_u^\top \mathbf{z}_v}}_{\text{embedding similarity}} - \underbrace{\boxed{\mathbf{A}}}_{\text{(weighted) adjacency matrix for the graph}}{}_{u,v} \|^2$$

Adjacency-based Similarity

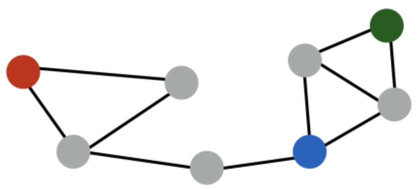
$$\mathcal{L} = \sum_{(u,v)\in V \times V} \left\| \mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v} \right\|^2$$

- Find embedding matrix $\mathbf{Z} \in \mathbb{R}^{d \times |V|}$ that minimizes the loss $\mathcal{L}$
  - Option 1: **Use stochastic gradient descent** (**SGD**) as a general optimization method.
    - Highly scalable, general approach
  - Option 2: **Solve matrix decomposition solvers** (e.g., SVD or QR **decomposition** routines).
    - Only works in limited cases.

Adjacency-based Similarity

$$\mathcal{L} = \sum_{(u,v) \in V \times V} \|\mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v}\|^2$$

- **Drawbacks**:
  - O(|V|2) runtime. (Must consider all node pairs.)
    - Can make O([E|) by only summing over non-zero edges and using regularization (e.g., Ahmed et al., 2013)
  - O(|V|) parameters! (One learned vector per node).
  - Only considers direct, local connections.

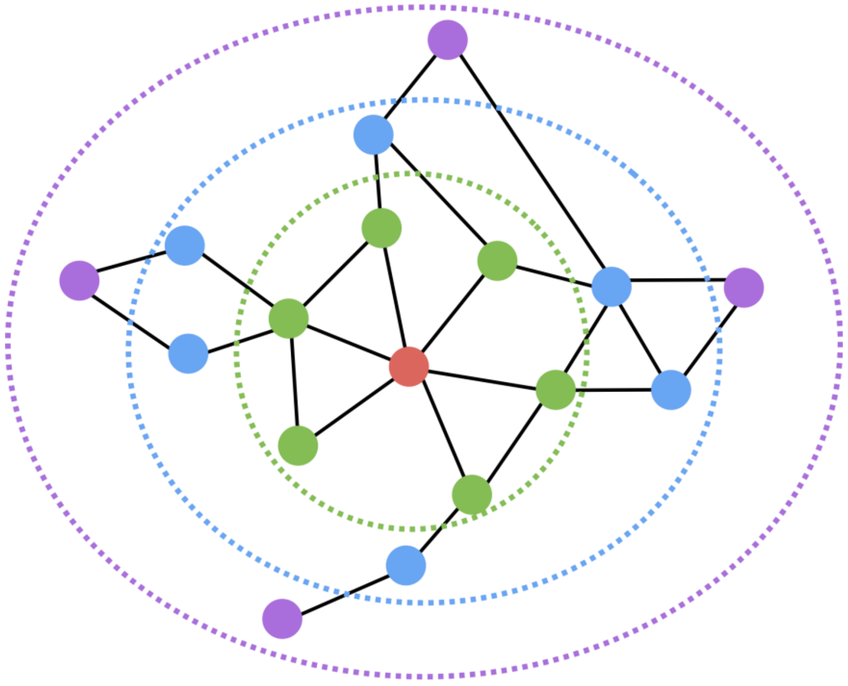


e.g., the blue node is obviously more similar to green compared to red node, despite none having direct connections.

2. Multi-hop Similarity

- Material based on:
  - Cao et al. 2015. GraRep: Learning Graph Representations with Global Structural Information (CIKM 2015)
  - Ou et al. Asymmetric Transitivity Preserving Graph Embedding. (KDD 2016)

- **Idea**: Consider k-hop node neighbors.
  - E.g., two or three-hop neighbors.



- **Red:** Target node
- **Green**: 1-hop neighbors
  - **A** (i.e., adjacency matrix)
- **Blue**: 2-hop neighbors
  - $\mathbf{A}^2$
- **Purple**: 3-hop neighbors
  - $\mathbf{A}^3$

- **Basic idea:**

$$\mathcal{L} = \sum_{(u,v) \in V \times V} \| \mathbf{z}_u^\top \mathbf{z}_v - \boxed{\mathbf{A}_{u,v}^k} \|^2$$

- Train embeddings to predict k-hop neighbors.
- In practice (GraRep from Cao et al, 2015):
  - Use log-transformed, probabilistic adjacency matrix:

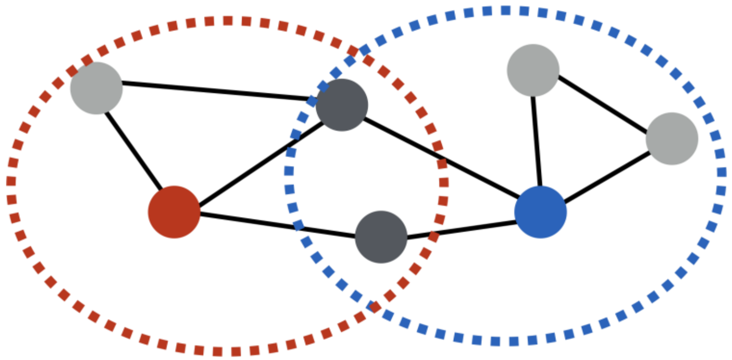
$$\tilde{\mathbf{A}}_{i,j}^k = \max\left(\log\left(\frac{(\mathbf{A}_{i,j}/d_i)}{\sum_{l \in V} (\mathbf{A}_{l,j}/d_l)^k} \right)^k - \alpha, 0 \right)$$

node degree　　constant shift

- Train multiple different hop lengths and concatenate output.

Multi-hop Similarity

- Another option: Measure overlap between node neighborhoods.



- Example overlap functions:
- Jaccard similarity
- Adamic-Adar score

Multi-hop Similarity

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \|\underbrace{\boxed{\mathbf{z}_u^\top \mathbf{z}_v}}_{\text{embedding similarity}} - \underbrace{\boxed{\mathbf{S}_{u,v}}}_{\text{multi-hop network similarity (i.e., any neighborhood overlap measure)}}\|^2$$

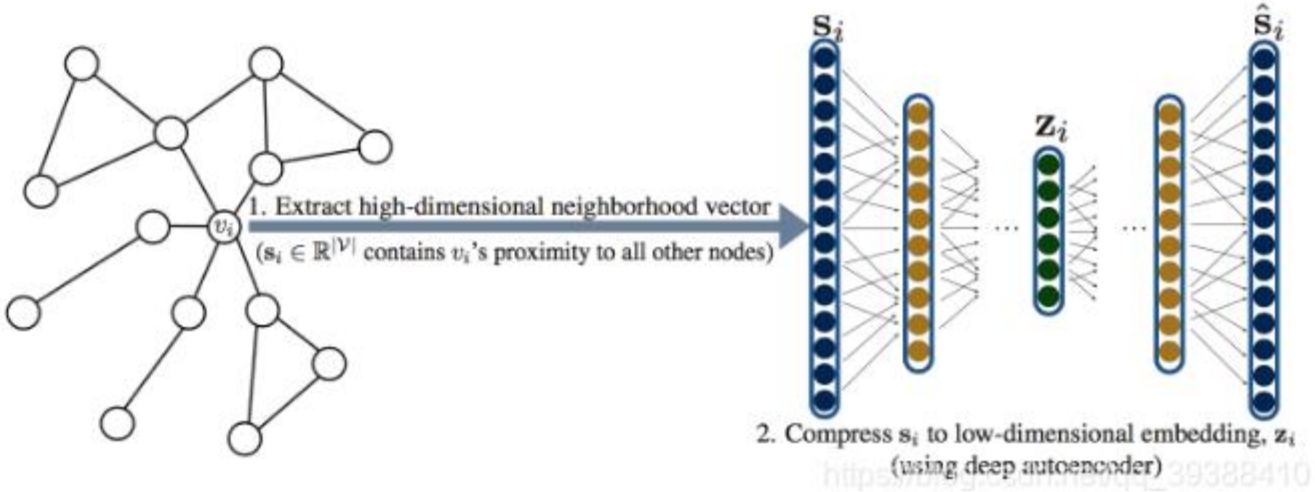
- $\mathbf{S}_{u,v}$ is the neighborhood overlap between u and v (e.g., Jaccard overlap or Adamic-Adar score).
- This technique is known as HOPE (Yan et al., 2016).

Summary

- Basic idea so far:
  1) Define pairwise node similarities.
  2) Optimize low-dimensional embeddings to approximate these pairwise similarities.
- Issues:
  - **Expensive**: Generally O(|V|2), since we need to iterate over all pairs of nodes.
  - **Brittle**: Must hand-design deterministic node similarity measures.
  - **Massive parameter space**: O(|V|) parameters

SDNE model

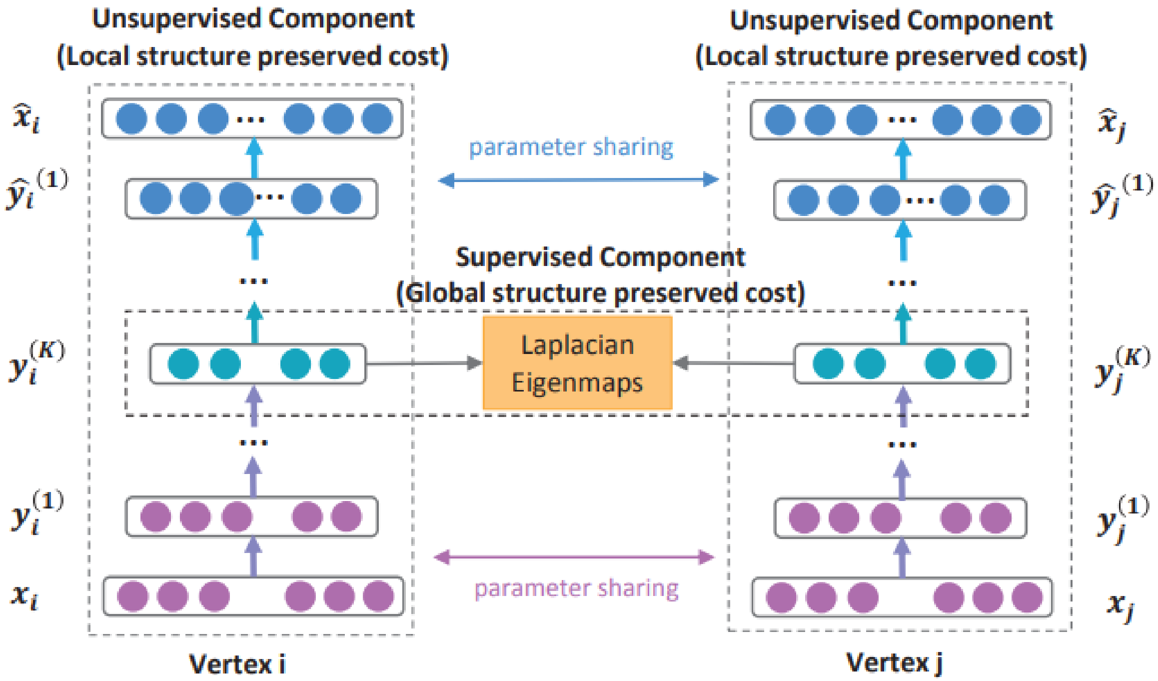
- The difference between SDNE and {Deepwalk, LINE, Node2vec} is that it is not based on the idea of random walks
- The main idea is based on Autoencoder to reduce the dimensionality of input vector and compress it, and then reconstruct the features.



https://blog.csdn.net/qq_39388410/article/details/103895874

SDNE model

- The framework of the semi-supervised deep model of SDNE.
- Similar to LINE, SDNE also wants to preserve 1st and 2nd order similarity and optimize at the same time to capture both local pairwise similarity and the similarity of the node neighborhood structure.



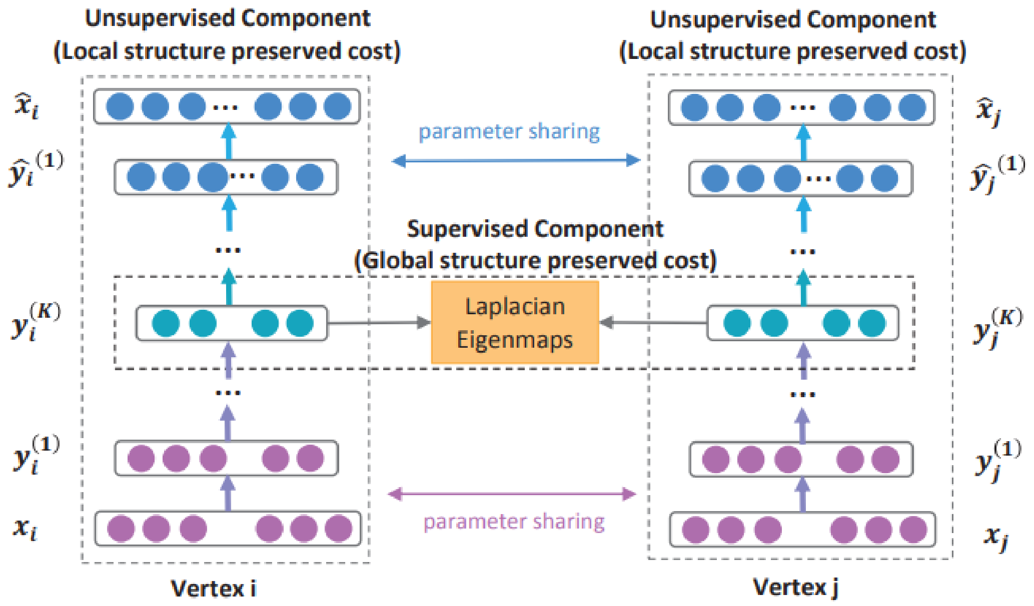
SDNE model

- Then given the input x_i, the hidden representations for each layer are:

$$\mathbf{y}_i^{(1)} = \sigma(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)})$$
$$\mathbf{y}_i^{(k)} = \sigma(W^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}), k = 2, ..., K$$

- The goal of the autoencoder is to minimize the reconstruction error of the output and the input.
- The loss function:

$$\mathcal{L} = \sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2$$



SDNE model

- Loss function for first-order proximity:

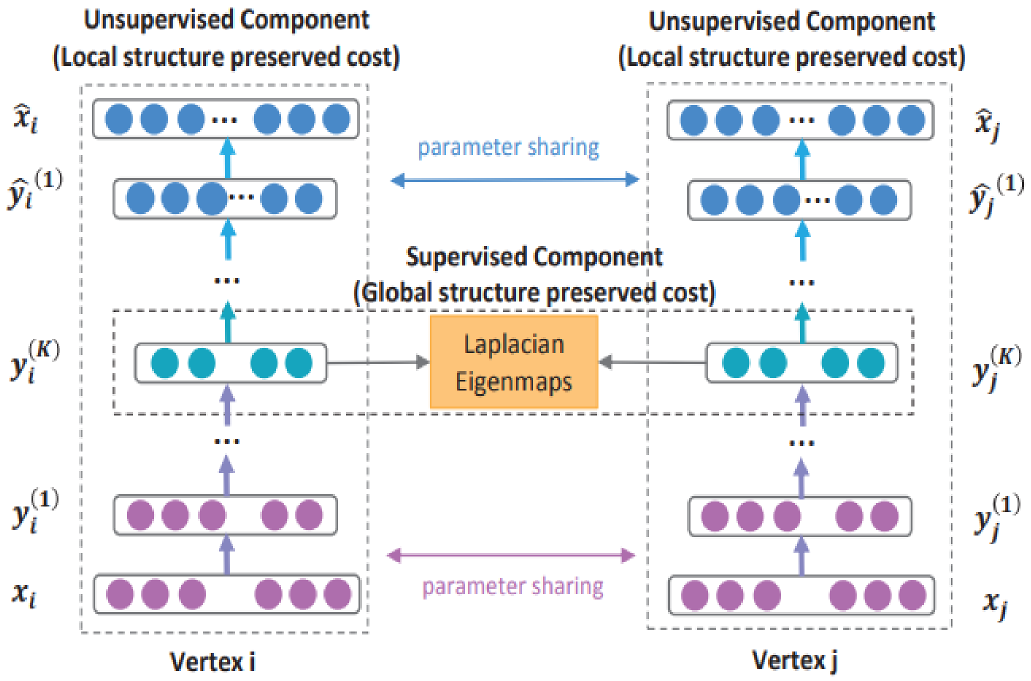
$$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)}\|_2^2$$

$$= \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2$$

- Impose more penalty to the reconstruction error of the non-zero elements than that of zero elements:

$$\mathcal{L}_{2nd} = \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b_i}\|_2^2$$

$$= \|(\hat{X} - X) \odot B\|_F^2$$

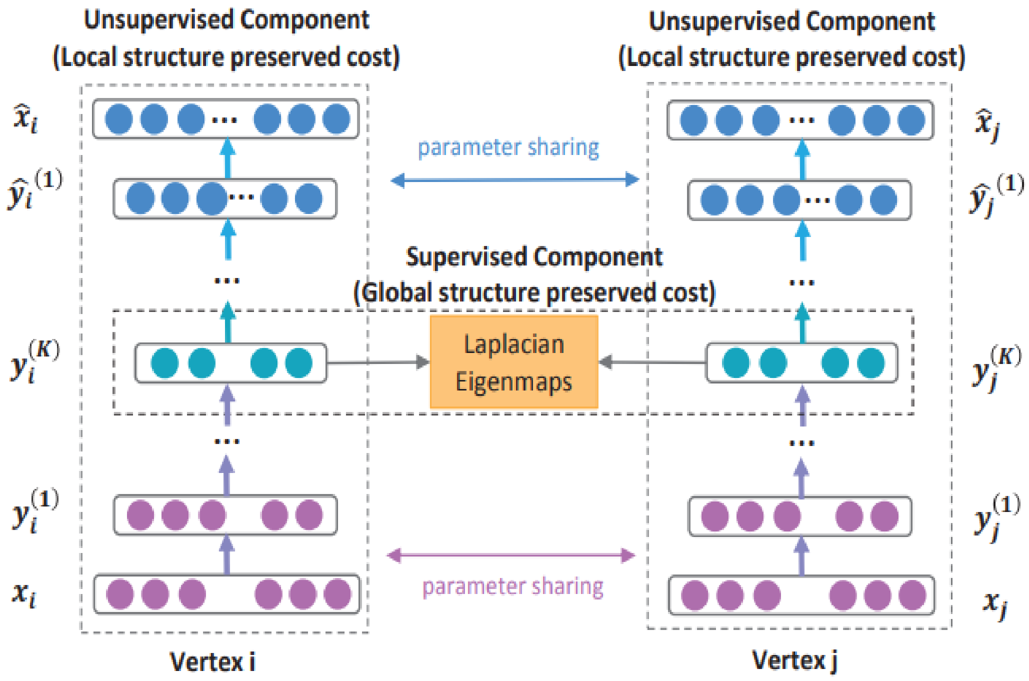


- To preserve the first-order and second-order proximity simultaneously, we need to minimize the joint loss:

$$\mathcal{L}_{mix} = \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{reg}$$
$$= \|(\hat{X} - X) \odot B\|_F^2 + \alpha \sum_{i,j=1}^{n} s_{i,j}\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 + \nu\mathcal{L}_{reg}$$

where Lreg is an L2-norm regularizer term to prevent overfitting, which is defined as follows:

$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{k=1}^{K}(\|W^{(k)}\|_F^2 + \|\hat{W}^{(k)}\|_F^2)$$



Subgraph2Vec: Motivation

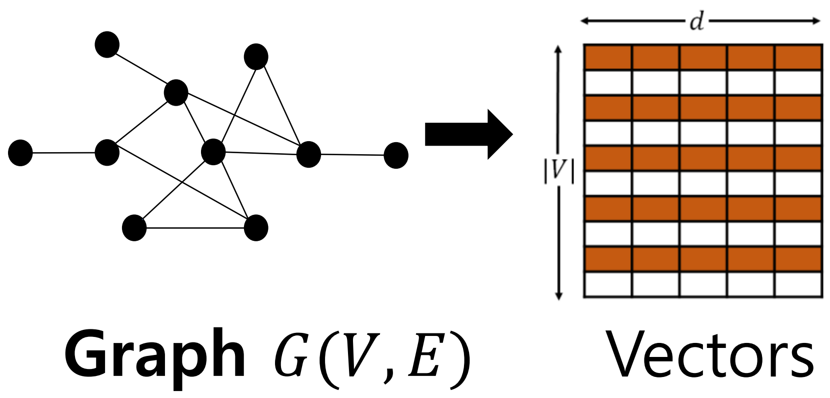
- Popular previous works include

DeepWalk[Perozzi+, KDD2014]

Node2vec[Grover+, KDD 2016]

SDNE[Wang+, KDD 2016]

LINE[Tang+,WWW 2015]



Limited just to the node embeddings

Subgraph2Vec model: Idea

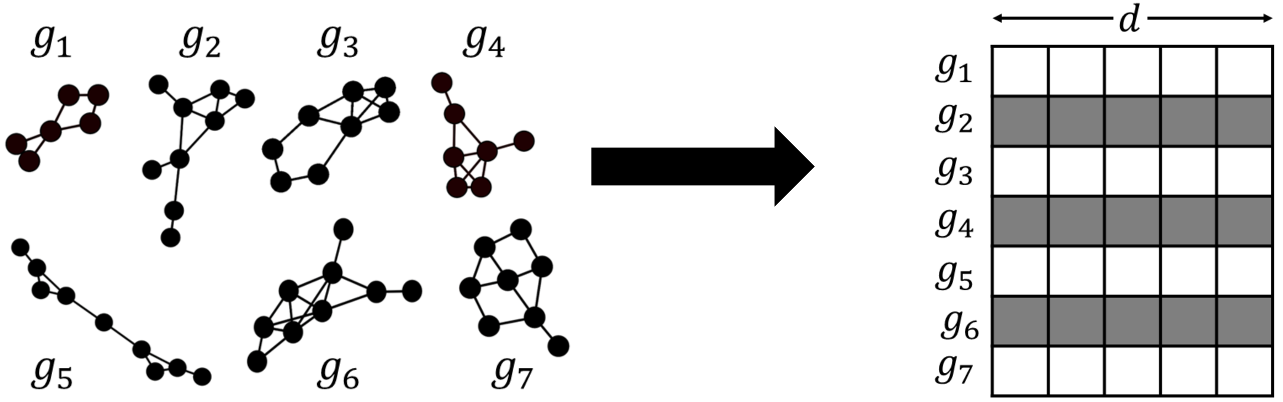
- Learning representation of substructures
  - Extend the WL relabeling strategy to define a proper context for a given subgraph.
  - A modification to the skipgram model enabling it to capture varying length radial contexts

Problem Formulation: Setting

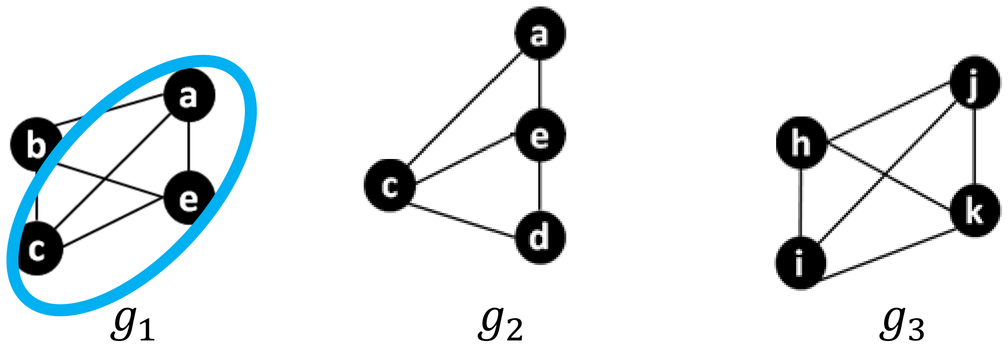
- **Given**:
  - A set S={g_1, g_2, …, g_n } of subgraphs
  - Typically for the same graph
  - An integer d
- **Learning**:
  - d-dimensional embedding for each subgraph
  - Such that pre-defined subgraph property is preserved

Set of Subgraphs

Subgraph Embedding



- What subgraph property to preserve?
  - Neighbourhood Property:
    - Captures neighbourhood information within the subgraph



- Subgraph g_1 and g_2 share neighbourhood
- Subgraph g_3 does not

Algorithm 1: Generate rooted subgraphs

- Generate rooted subgraphs around every node in a given graph

- Considers all the rooted subgraphs (up to a certain degree) of neighbours of r as the context of target subgraphs

Algorithm 2: GETWLSUBGRAPH (v, G, d)

input : v: Node which is the root of the subgraph
$G = (V, E, \lambda)$: Graph from which subgraph has to be extracted
d: Degree of neighbours to be considered for extracting subgraph

output: $sg_v^{(d)}$: rooted subgraph of degree d around node v

begin
  $sg_v^{(d)} = \{\}$
  if $d = 0$ **then**
    $sg_v^{(d)} := \lambda(v)$
  else
    $\mathcal{N}_v := \{v' \mid (v, v') \in E\}$
    $M_v^{(d)} := \{\text{GETWLSUBGRAPH}(v', G, d-1) \mid v' \in \mathcal{N}_v\}$
    $sg_v^{(d)} := sg_v^{(d)} \cup \text{GETWLSUBGRAPH}(v, G, d-1) \oplus sort(M_v^{(d)})$
  return $sg_v^{(d)}$

Algorithm 2: Learn embeddings of those subgraphs

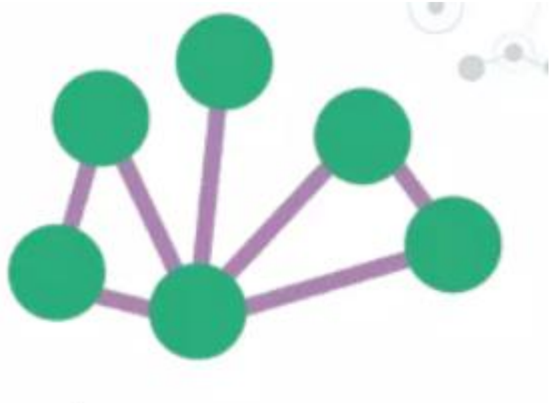
- The skipgram model maximizes co-occurrence probability among the sub-graphs that appear within a given context window.

Algorithm 3: RADIALSKIPGRAM $(\Phi, sg_v^{(d)}, G, D)$

begin
- $context_v^{(d)} = \{\}$
- **for** $v' \in$ NEIGHBOURS(G, v) **do**
  - **for** $\partial \in \{d-1,\ d,\ d+1\}$ **do**
    - **if** $(\partial \geq 0$ *and* $\partial \leq D)$ **then**
      - $context_v^{(d)} = context_v^{(d)} \cup$ GETWLSUBGRAPH(v', G, ∂)
- **for** *each* $sg_{cont} \in context_v^{(d)}$ **do**
  - $J(\Phi) = -\log \Pr\left(sg_{cont} | \Phi(sg_v^{(d)})\right)$
  - $\Phi = \Phi - \alpha \frac{\partial J}{\partial \Phi}$

2. Homogeneous vs Heterogeneous Graphs

- Single node type and single edge type
  - E.g.,
    - Users **follow** other Users
- Heterogeneous Graphs
  - Multiple node and/or edge types
  - E.g.,
    - Users follow other Users
    - Users fave tweets
    - Users reply to tweets



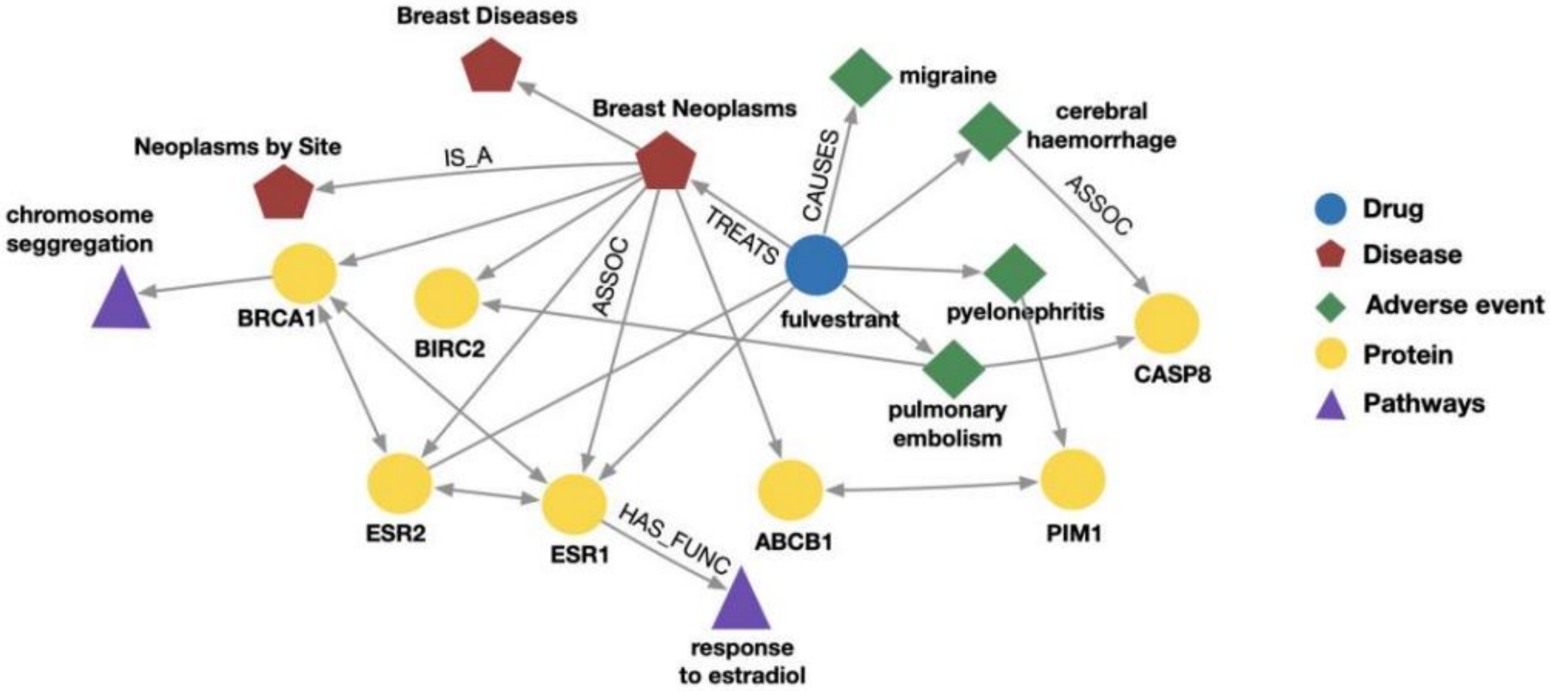
- A heterogeneous graph is defined as:

$$G = (V, E, R, T)$$

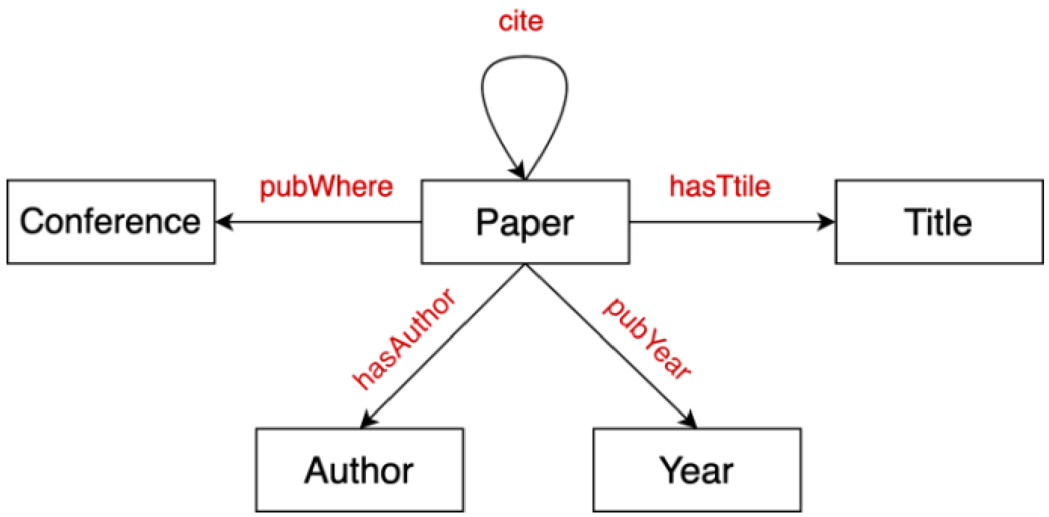
- Nodes with node types $v_i \in V$
- Edges with relation types $(v_i, r, v_j) \in E$
- Node type $T(v_i)$
- Relation type $r \in R$

Heterogeneous Graphs: Examples

- Biomedical Knowledge Graphs
  - Example node: Migraine
  - Example edge: (fulvestrant, Treats, Breast Neoplasms)
  - Example node type: Protein
  - Example edge type (relation): Causes

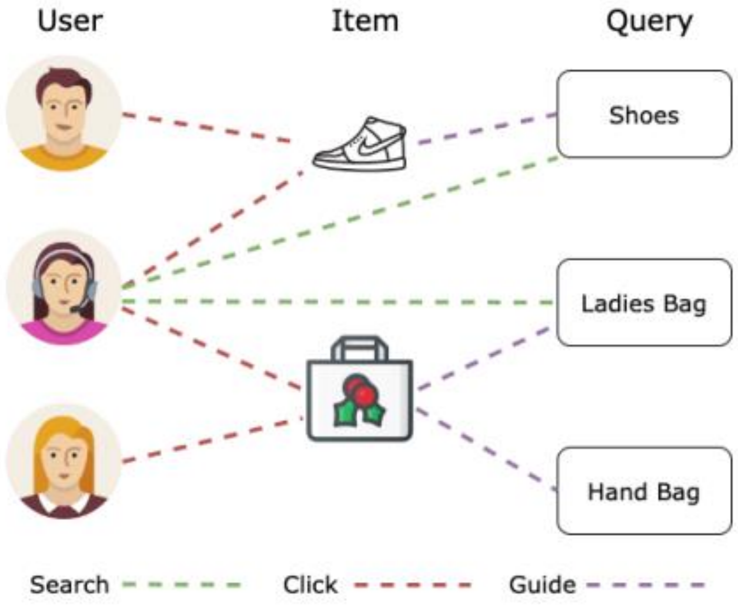


- Academic Graphs:
  - Example node: ICML
  - Example edge: (GraphSAGE, NeurIPS)
  - Example node type: Author
  - Example edge type (relation): pubYear



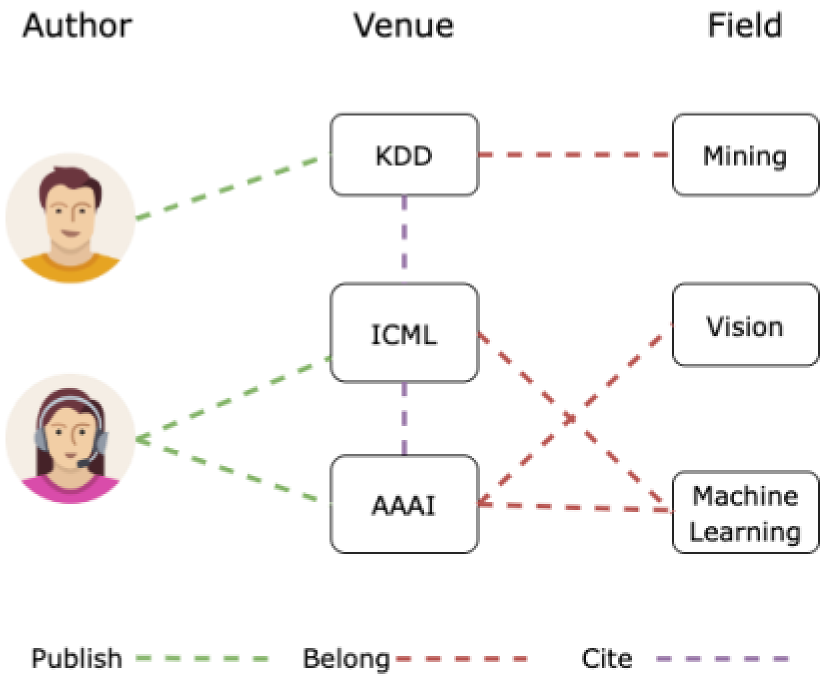
Many Graphs are Heterogeneous Graphs

- Example: E-Commerce Graph
- Node types: User, Item, Query, Location, ...
- Edge types: Purchase, Visit, Guide, Search, …
- Different node type's features spaces can be different!



Many Graphs are Heterogeneous Graphs

- Example: Academic Graph
- Node types: Author, Paper, Venue, Field, ...
- Edge types: Publish, Cite, …
- Benchmark dataset: Microsoft Academic Graph



Why can't we use homogeneous learning methods?

- **Complex Structure**
  - The structure in Heterogeneous Graphs is highly semantic-dependent, such as a meta-path structure
- **Heterogeneous Attributes**
  - different types of nodes and edges have different attributes which are located in different feature spaces.
  - To effectively fuse the attributes of neighbors Heterogeneous methods have to overcome this heterogeneity.

Meta path [Han VLDB'11]

- A sequence of node class sets connected by edge types

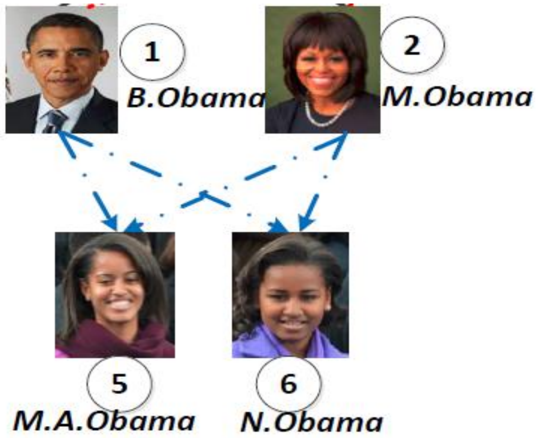
$$\Pi^{1\ldots n} = \mathsf{C}_1 \xrightarrow{\mathsf{e}_1} \ldots \mathsf{C}_i \xrightarrow{\mathsf{e}_i} \ldots \mathsf{C}_n$$

- Benefits of Meta Paths
- Multi-hop relationships instead of direct links
- Combine multiple relationships

$$m1: \mathsf{USPresident} \xrightarrow{\mathsf{hasChild}} \mathsf{Person} \xrightarrow{\mathsf{hasChild}^{-1}} \mathsf{USFirstLady},$$

$$m2: \mathsf{USPresident} \xrightarrow{\mathsf{memberOf}} \mathsf{USPoliticalParty} \xrightarrow{\mathsf{memberOf}^{-1}} \mathsf{USFirstLady},$$

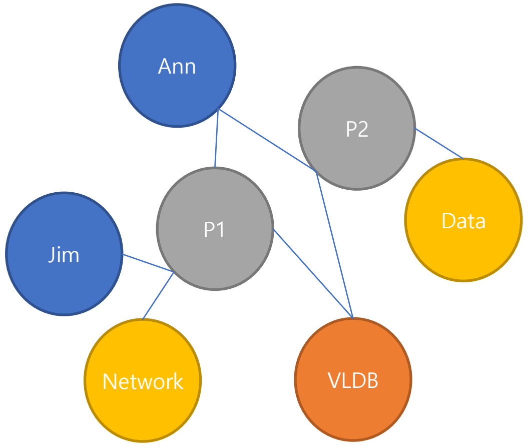
$$m3: \mathsf{USPresident} \xrightarrow{\mathsf{citizenOf}} \mathsf{Country} \xrightarrow{\mathsf{citizenOf}^{-1}} \mathsf{USFirstLady}.$$



Similarity score

- Similarity score for a node pair following a single meta-path
  - **Path Count** (PC) [Han ASONAM'11]
    - Number of the paths following a given meta-path
  - **Path Constrained Random Walk** (PCRW) [Cohen KDD'11]
    - Transition probability of a random walk following a given meta-path
- Similarity score for a node pair following a combination of multiple meta-paths
  - Aggregate Function F to combine the similarity scores for each single meta path

- **Meta Path**
  - Two objects can be connected via different connectivity paths
  - E.g., two authors can be connected by
    - “author-paper-author” (APA)
    - “author-paper-author-paper-author” (APAPA)
    - “author-paper-venue-paper-author” (APCPA)
- Each connectivity path represents a different semantic meaning and implies different similarity semantics

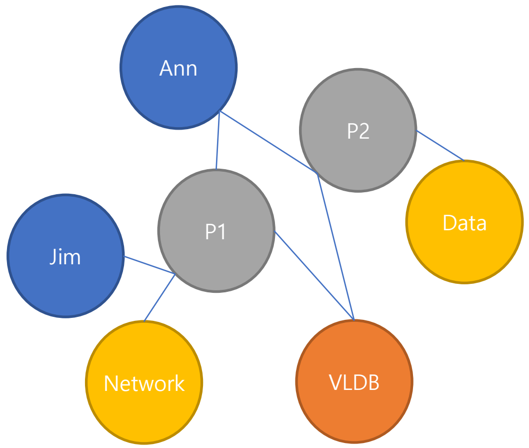


Meta Paths Generation

- A meta path is a meta level description of the topological connectivity between objects
  - Given a Network Schema, A meta path can be defined as

$$A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} A_{l+1}$$

  - Can be considered as a new relation defined on type A_1 and A_{l+1}



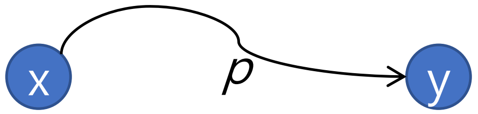
Conventional Similarity measures

- Path Count:
  - The number of path instances p between x and y following P:

$$s(x,y) = |\{p : p \in P\}|$$

- Random Walk:
  - The probability *Prob(p)* of the random walk that starts from x and ends with y following meta path *P*, which is the sum of the probabilities of all the path instances *p*

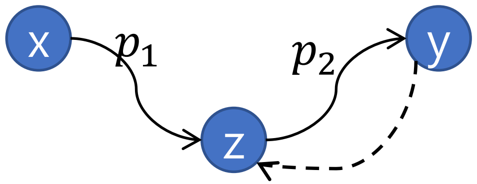
$$s(x,y) = \sum_{p \in P} \mathrm{Prob}(p)$$



Conventional Similarity measures (2)

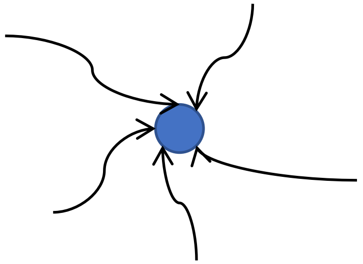
- Pairwise Random Walk
  - For a meta path P that can be decomposed into two shorter meta paths with the same length P = (P_1P_2), pairwise random walk probability is the probabilities starting from x and y and reaching the same middle object z

$$s(x,y) = \sum_{(p_1p_2)\in(P_1P_2)} \Pr ob(p_1) \Pr ob(p_2^{-1})$$

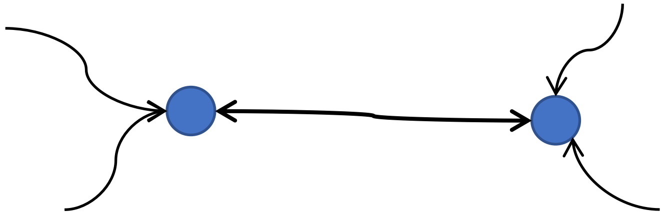


PathSim: A Novel Meta Path-Based Similarity Measure

- Similarity in terms of ‘Peers’
  - Two similar peer object should not only be strongly connected, but also share comparable visibility.
- Path count and Random walk (RW)
  - Favor highly visible objects (objects with large degrees)



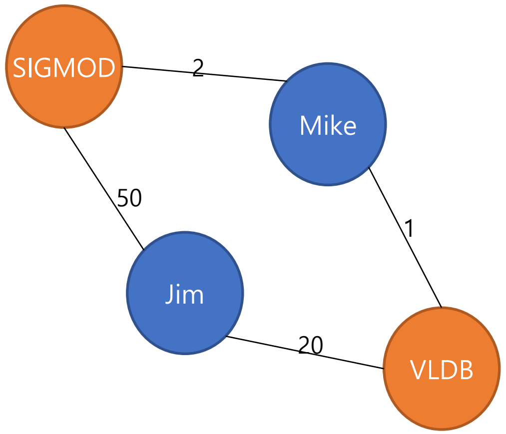
- Pairwise random walk (PRW)
  - Favor pure objects (objects with highly skewed scatterness in their in-links or out-links)
- PathSim
  - Favor “peers” (objects with similar visibility and strong connectivity under the given meta path)



PathSim: A Novel Meta Path-Based Similarity Measure (2)

- Restricted on Round-Trip Meta Path
  - A round-trip meta path is a path of the form of P = $\left(P_l P_l^{-1}\right)$
  - Guarantees a symmetric relation

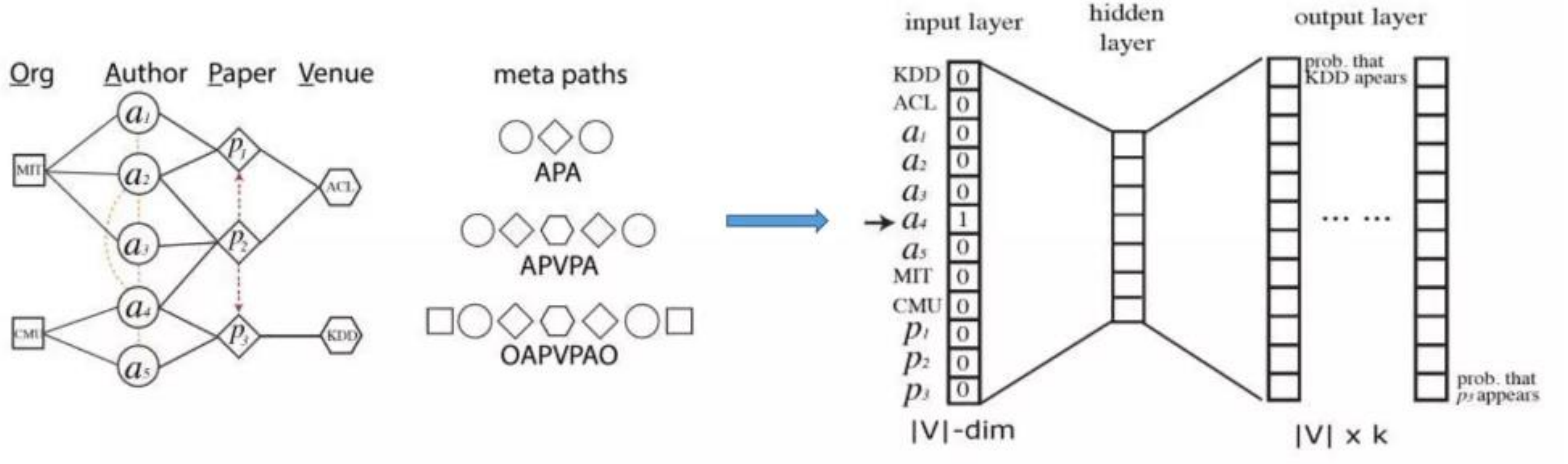
$$s(x, y) = \frac{2 \times |\{p_{x \rightsquigarrow y} : p_{x \rightsquigarrow y} \in \mathcal{P}\}|}{|\{p_{x \rightsquigarrow x} : p_{x \rightsquigarrow x} \in \mathcal{P}\}| + |\{p_{y \rightsquigarrow y} : p_{y \rightsquigarrow y} \in \mathcal{P}\}|}$$



s(Mike, Jim) = $\frac{2*(2*50+1*20)}{(2*2+1*1)+(50*50+20*20)} = 0.0826$

Metapath2vec/ Metapath2vec++

- A meta-path is a sequence of node types encoding key composite relations among the involved node types
- Meta-paths are used to guide random walks to redefine the neighborhood of a node
- Metapath2Vec (KDD 2017)

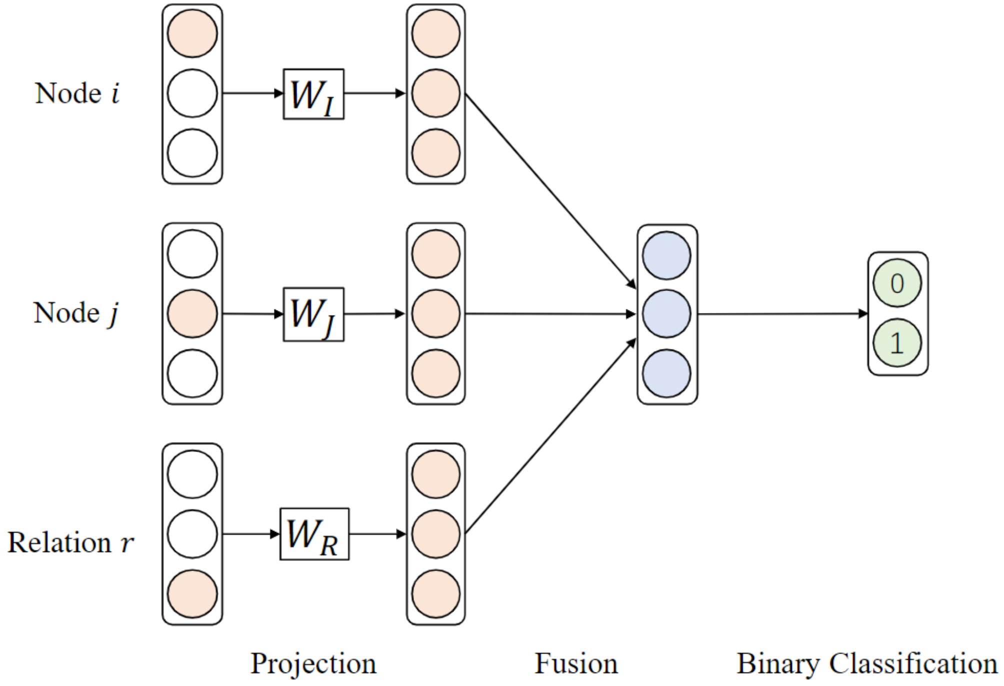


- Metapath2vec++ samples the negative nodes of the same type as the central node by maintaining separate multinomial distributions for each node type in the output layer of the skip-gram model

Yuxiao Dong , et. Al., metapath2vec: Scalable Representation Learning for Heterogeneous Networks (KDD 2017)

HIN2vec (Fu et.al, CIKM 2017)

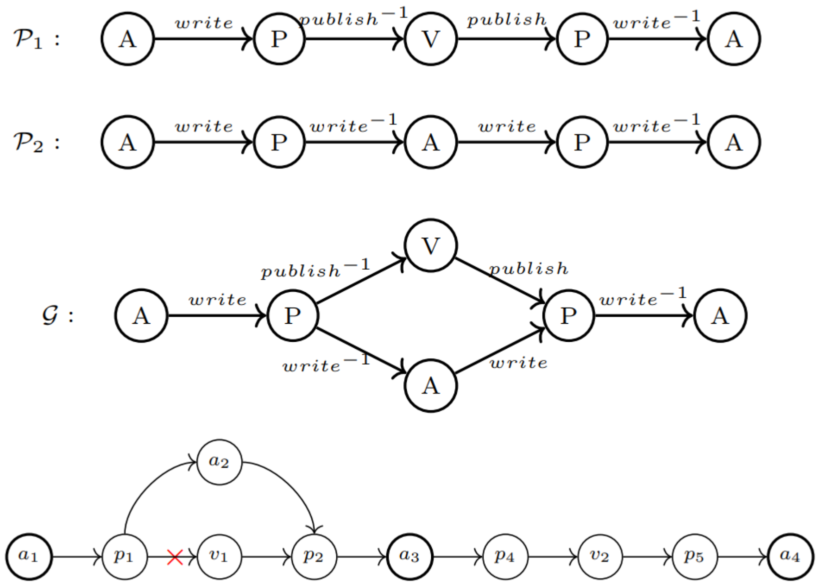
- Combines first-order relation and high-order relation (i.e. meta-paths)
- HIN2vec works in a multi-label classification style by predicting whether two given nodes are connected by a meta-path



Fu, T. HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning. *Proceedings of the 2017 ACM on CIKM*

MetaGraph2Vec

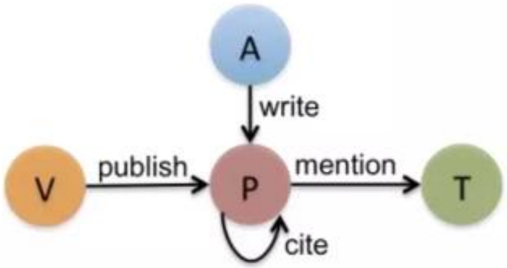
- A meta-graph is a DAG defined on the given HIN schema which has only a single source node and a single target node
- Real-world HINs often have to deal with sparse or missing connections. As the following example shows, meta-paths P_1 and P_2 will fail to capture path $a_1 \rightarrow a_4$ the highlighted link is missing.
- However, the meta-graph G provides a richer structural context and is able to perform this random walk. This shows the meta-graph's capability to match more paths in a sparse context.

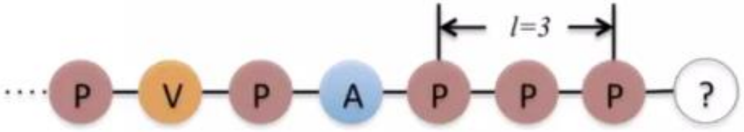


JUST – Heterogeneous Graph embedding technique

- **Challenges**:
  - How to select meta-paths?
    - Graph specific and highly depends on prior knowledge from domain experts.
    - Strategies to combine a set of meta-paths can be complex and computationally expensive
  - The choice of metapaths highly affects the quality of the learned node embeddings for a specific task.
- **Are metapaths necessary?**

- **JUST idea:**
  - Random walk with JUmp and Stay strategies to probabilistically control the random walk
  - Learn node embeddings with SkipGram model
- Jump or Stay?
  - **Objective**: Balance the number of heterogeneous a traversed during random walks.





$$\Pr_{stay}(v_i) = \begin{cases} 0, \text{ if } V_{stay}(v_i) = \varnothing \\ 1, \text{ if } (V^{q}_{stay}(v_i) \mid q \in Q, q \neq \phi(v_i)\} = \varnothing \\ \alpha^{l}, \text{ otherwise} \end{cases}$$

Where:

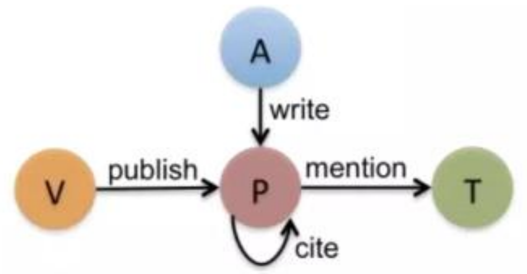
$\alpha \in [0,1]$ is an initial stay probability

l refes to the number of nodes consecutively visited in the same domain

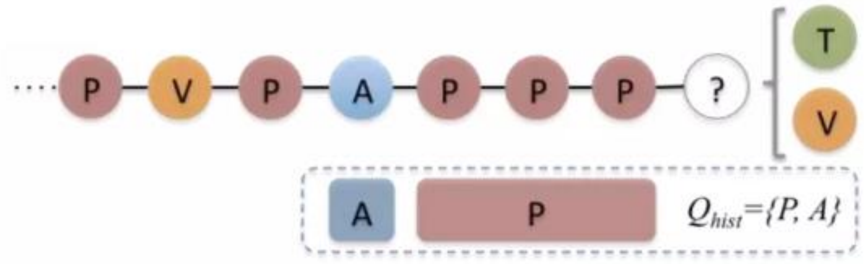
Where to JUmp?

- **Objective**: control the randomness in choosing a target domain
- Define a fixed length queue Q_{hist} to memorize up to m previously visited domains:

$$Q_{Jump}(v_i) = \begin{cases} \{q \mid q \in Q \wedge q \notin Q_{hist}, V^{q}_{jump}(v_i) \neq \emptyset\}, \text{ if not empty} \\ \{q \mid q \in Q, q \neq \phi(v_i), V^{q}_{jump}(v_i) \neq \emptyset\}, \textit{ otherwise} \end{cases}$$



- For each node in the graph, initialize a random walk, until the maximum lenth is reached.
- Maximize the co-coccurance probability of two nodes appearing within a context window in the random walk using SkipGram model



Limitations of Meta-paths

- Meta-paths have to be manually customized based on task and dataset, hence requiring domain knowledge.
- They fail to capture more complex relationships such as motifs.
  - i.e. patterns of interconnections occurring in complex networks at numbers that are significantly higher than those in randomized networks4.
- The usage of meta-path is limited to the discrete space.
  - If two vertices are not structurally connected in the graph, metapath-based methods cannot capture their relations.

네트워크 과학 연구실 NETWORK SCIENCE LAB

가톨릭대학교 THE CATHOLIC UNIVERSITY OF KOREA